ИНТЕЛЛЕКТУАЛЬНЫЙ МЕГАПОЛИС

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

Инженерный класс в московской школе

НАПРАВЛЕНИЕ АВИАСТРОИТЕЛЬНЫЕ КЛАССЫ

ПРАКТИЧЕСКИЙ ЭТАП

МОСКВА
2025

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ РАЗРАБОТАНЫ:

Ткаченко Данилом Александровичем – Преподаватель Центра Детский технопарк МАИ «Траектория взлёта»

Бернацким Кириллом Сергеевичем – Ассистент кафедры 701 и Инженер НИО-701

Потешным Кириллом Александровичем - Инженер НИО-701 МАИ

Куяновым Владиславом Вячеславовичем - Инженер НИО-701 МАИ

Потешным Кириллом Александровичем - инженер НИО-701 МАИ

Куяновым Владиславом Вячеславовичем - инженер НИО-701 МАИ

МОСКВА
2025

Спецификация конкурсных материалов для проведения практического этапа Московского конкурса межпредметных навыков и знаний «Интеллектуальный мегаполис. Потенциал» в номинации «Инженерный класс» по направлению «Авиастроительные классы»

1. Назначение конкурсных материалов

Материалы практического этапа Московского конкурса межпредметных навыков и знаний «Интеллектуальный мегаполис. Потенциал» (далее – Конкурс) предназначены для оценки уровня практического подготовки участников Конкурса.

2. Условия проведения

Практического этап Конкурса проводится в очной дистанционной форме. При выполнении работы обеспечивается строгое соблюдение порядка организации и проведения Конкурса.
Задания практического этапа Конкурса выполняются с использованием следующего программного обеспечения (ПО):

1. Для выполнения заданий по кейсу №1 (программирование) - ПК, Python 3.9 и выше, PyCharm (Wing или VSCode);
2. Для выполнения заданий по кейсу №2 (3D-моделирование и 3D-печать) - системы автоматизированного проектирования (САПР) на выбор: T-Flex CAD 17, Компас-3D.

3. Продолжительность выполнения

На выполнение заданий практического этапа Конкурса отводится 120 минут. Во время проведения мероприятия участник может выйти из зоны проведения мероприятия не более чем на 5 минут, предупредив проктора на камеру. Мероприятие не продлевается на время отсутствия участника.

4. Содержание и структура

Индивидуальный вариант участника включает 2 независимых кейса, базирующихся на содержании элективных курсов «Программирование», «3D-моделирование и 3Dпечать».

5. Система оценивания

Максимальный балл за выполнение кейса – 60 баллов. Участник выбирает для решения только один кейс из двух. Для получения максимального балла за практический этап Конкурса необходимо верно выполнить все задания выбранного кейса.

6. Приложения

1. План конкурсных материалов для проведения практического этапа Конкурса.
2. Демонстрационный вариант конкурсных заданий практического этапа Конкурса.

Приложение 1

План конкурсных материалов для проведения практического этапа Московского конкурса межпредметных навыков и знаний «Интеллектуальный мегаполис. Потенциал» в номинации «Инженерный класс» по направлению «Авиастроительные классы»

№ задания	**Уровень сложности**	***Уникальные кодификаторы Конкурса***	**Контролируемые требования к проверяемым умениям**	**Балл**
1.1	*базовый*	*1.3. Основные типы данных; 2.2. Объявление переменных; 2.4. Оператор ввода; 2.5. Оператор вывода; 5.4. Понятие двумерного массива*	*Объявление переменных, работа с вводом/выводом, формирование двумерного массива*	*6*
1.2	*базовый*	*3.1. Условный оператор; 3.2. Операторы сравнения; 8.1. Объявление функции; 8.2. Определение функции; 8.3. Аргументы и параметры; 8.5. Вызов функции*	*Реализация функции проверки простого числа (алгоритмическая сложность, условные операторы)*	*6*
1.3	*повышенный*	*4.1. Цикл с предусловием; 4.4. Счётчик цикла; 6.1. Сортировка пузырьком*	*Самостоятельная реализация пузырьковой сортировки*	*8*
1.4	*сложный*	*5.3. Обработка и поиск в одномерном массиве; 5.4. Понятие двумерного массива; 8.4. Возвращаемое значение*	*Поиск подматрицы по заданным условиям, использование вложенных циклов*	*15*
1.5	*сложный*	*2.5. Оператор вывода; 8.1. Объявление функции; 8.2. Определение функции; 8.3. Аргументы и параметры; 8.5. Вызов функции*	*Сборка программы: обработка матрицы, вызов функций, консольное взаимодействие*	*15*
1.6	*базовый*	*1.3. Основные типы данных; 8.2. Определение функции;*	*Расчёт суммы и среднего*	*5*

		8.4. Возвращаемое значение	*арифметического простых чисел, округление результата*	
1.7	*базовый*	*3.1. Условный оператор;* *8.5. Вызов функции*	*Реализация сообщения о невозможности выполнения при отсутствии простых чисел*	*5*
			Итоговый балл	***60***

Критерии снижения баллов выполненных заданий

№	Задание	Ошибка	Снижаемый балл
1	Задание №1.1	Неправильное формирование матрицы (например, неиспользование вложенных списков)	3
2		Нарушение логики ввода/вывода (например, игнорирование форматированного вывода)	3
3	Задание №1.2	Ошибки в реализации is_prime (например, неэффективный перебор, проверка до n)	3
4	Задание №1.3	Использование встроенной сортировки вместо пузырьковой	4
5		Ошибка в реализации пузырьковой сортировки (например, пропущен проход или проверка)	4
6	Задание №1.4	Неверное определение границ ядра (ошибка в индексах строк/столбцов)	5

7		Необоснованное повторное вычисление простоты чисел в ядре	3
8	Задание №1.5	Отсутствие модульной структуры (все в main, без вызова функций)	5
9		Ошибки в организации взаимодействия с пользователем	5
10	Задание №1.6	Ошибка в расчёте суммы/среднего значения	3
11		Отсутствие округления результата до двух знаков	2
12	Задание №1.7	Отсутствует проверка на наличие простых чисел и соответствующее сообщение	3
13		Некорректное условие при проверке пустого списка простых чисел	2
14	Задание №2.1	Несоответствие геометрии (скругление вместо фаски; сфера вместо цилиндра; отсутствие элементов и пр.).	4
15		Несоответствие размера (ошибки в номинальных значениях или их принадлежности).	3
16	Задание №2.2	Несоответствие геометрии (скругление вместо фаски; сфера вместо цилиндра; отсутствие элементов и пр.).	3
17		Несоответствие размера (ошибки в номинальных значениях или их принадлежности).	2
18		Несоответствие размера (размер сборки отличаются от требуемого).	3

19	Задание №2.3	Не хватает сборочной единицы.	3
20		Сопряжение установлено некорректно (детали имеют взаимопроникновение или относительную степень свободы) или отсутствует.	2

Приложение 2

Демонстрационный вариант конкурсных заданий практического этапа Московского конкурса межпредметных навыков и знаний «Интеллектуальный мегаполис. Потенциал» в номинации «Инженерный класс» по направлению «Авиастроительные классы»

Кейс №1 «Программирование»

«Ядро простых чисел в матрице»

Разработать на языке Python программу для поиска ядра простых чисел в двумерной матрице.

Пользователь вводит размеры матрицы n и m, затем n × m целых чисел. Требуется:
1. Сформировать матрицу n×m из введённых чисел;
2. Найти все простые числа в матрице;
3. Определить минимальный прямоугольный подмассив ("ядро"), содержащий все строки и столбцы, где есть хотя бы одно простое число;
4. Вывести:
- Координаты ядра: (верхняя строка, левый столбец) до (нижняя строка, правый столбец);
- Все простые числа внутри ядра, отсортированные по возрастанию методом пузырьковой сортировки;
- Сумму и среднее значение найденных простых чисел, округлённое до двух знаков после запятой;
5. Если в матрице нет простых чисел - вывести сообщение: «Простых чисел не найдено.»

Пример ввода:

3 4

4 5 6 7

8 9 10 11

13 14 15 16

Пример вывода:

Ядро простых чисел: (0, 1) до (2, 3)

Простые числа ядра (отсортированы): [5, 7, 11, 13]

Сумма: 36

Среднее значение: 9.00

Таблица с функциями:

Функция	Параметры	Описание	Пример вызова функции
is_prime	число: **int**	Проверка числа на простоту	is_prime(13)
bubble_sort	массив (список): **list[int]**	Сортировка массива методом пузырька	bubble_sort([5, 3, 8])
find_prime_core	двумерный массив(список): **list[list[int]]**	Нахождение ядра простых чисел	find_prime_core(matrix)
print_core_info	список простых чисел: **list[int]**, список координат левого верхнего угла: **list[int, int]**, список координат правого нижнего угла: **list[int, int]**	Форматированный вывод результата по примеру	print_core_info(primes, top_left, bottom_right)

Указания для задания:

- Указание типов входных параметров в явном виде необязательно
- Предполагается, что программе на вход подаются только корректные (не вызывающие ошибок) последовательности команд.
- Размеры матрицы n и m - целые положительные числа от 1 до 100 включительно
- Элементы матрицы - целые положительные числа (неотрицательные и ноль не используются)
- Все значения вводятся пользователем с клавиатуры:
  - Сначала - два числа через пробел (n m);
  - Затем - построчный ввод n строк по m чисел в каждой.
- Проверка на простое число:
  - Проверка реализуется в виде отдельной функции is_prime(n: int) -> bool.
  - Допустима реализация через деление на все числа до √n.
  - Использование готовых библиотек (например, sympy) не допускается.
- Поиск координат ядра:
  - Найти индексы всех строк, содержащих хотя бы одно простое число.
  - Найти индексы всех столбцов, содержащих хотя бы одно простое число.
  - Использовать минимальные и максимальные индексы для определения прямоугольной подматрицы.

- Поиск координат ядра:
  - Из ядра (подматрицы) необходимо извлечь все простые числа.
  - Эти числа собираются в отдельный список **list[int]**.
- Сортировка:
  - Отсортировать список primes_in_core необходимо методом пузырьковой сортировки.
  - Запрещено использовать .sort() или sorted().
  - Сортировка реализуется вручную **def bubble_sort(lst: list[int]) -> list[int]**
- Обработка ошибок:
  - Если в матрице нет ни одного простого числа вывести сообщение “ Простых чисел не найдено”.
- Функция find_prime_core:
  - Возвращает три значения: 1) отсортированный список по неубыванию простых чисел; 2) список координат левого верхнего угла; 3) список координат правого нижнего угла;
  - Если нет строк и столбцов, где содержатся простые числа вернуть значения: None, None, None

Интерфейс для работы с пользователем:

Интерфейс должен быть реализован в виде текстовых команд, вводимых пользователем. Примерная структура интерфейса:
def main():

```
n, m = map(int, input("Введите размеры матрицы: ").split())

print("Введите элементы матрицы построчно:")
```

А дальше необходимо реализовать корректный ввод данных, а также вызов соответствующей функции и обработки её результата.

Примеры Ввода и вывода программы
Пример 1: Несколько простых чисел, ядро 3×3

Ввод:

3 4

4 5 6 7

8 9 10 11

13 14 15 16

Вывод:

Ядро простых чисел: (0, 1) до (2, 3)

Простые числа ядра (отсортированы): [5, 7, 11, 13]

Сумма: 36

Среднее значение: 9.00

Пример 2: В матрице нет ни одного простого числа

Ввод:

2 3

4 6 8

9 10 12

Вывод:

Простых чисел не найдено.

Пример 3: Один простой элемент в центре матрицы

Ввод:

3 3

4 6 8

9 13 10

14 15 16

Вывод:

Ядро простых чисел: (1, 1) до (1, 1)

Простые числа ядра (отсортированы): [13]

Сумма: 13

Среднее значение: 13.00

Пример 4: Простые только в одном столбце

Ввод:

3 4

2 4 6 8

3 9 10 12

5 14 15 16

Вывод:

Ядро простых чисел: (0, 0) до (2, 0)

Простые числа ядра (отсортированы): [2, 3, 5]

Сумма: 10

Среднее значение: 3.33

Пример 5: Простые в разных строках и столбцах, пересекаются не полностью

Ввод:

4 4

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

Вывод:

Ядро простых чисел: (0, 1) до (3, 2)

Простые числа ядра (отсортированы): [2, 3, 5, 7, 11, 13]

Сумма: 41

Среднее значение: 6.83

Пример 6: Простой элемент в одном углу (верхний левый угол)

Ввод:

3 3

2 4 6

8 10 12

14 15 16

Вывод:

Ядро простых чисел: (0, 0) до (0, 0)

Простые числа ядра (отсортированы): [2]

Сумма: 2

Среднее значение: 2.00

Пример 7: Матрица полностью состоит из простых чисел

Ввод:

2 3

2 3 5

7 11 13

Вывод:

Ядро простых чисел: (0, 0) до (1, 2)

Простые числа ядра (отсортированы): [2, 3, 5, 7, 11, 13]

Сумма: 41

Среднее значение: 6.83

Пример 8: Только один элемент в матрице, и он простой

Ввод:

1 1

5

Вывод:

Ядро простых чисел: (0, 0) до (0, 0)

Простые числа ядра (отсортированы): [5]

Сумма: 5

Среднее значение: 5.00

Пример 9: Только один элемент в матрице, и он не простой

Ввод:

1 1

4

Вывод:

Простых чисел не найдено.

Пример 10: Простые числа на границах матрицы

Ввод:

4 4

2 4 6 3

8 9 10 11

12 13 14 15

17 18 19 23

Вывод:

Ядро простых чисел: (0, 0) до (3, 3)

Простые числа ядра (отсортированы): [2, 3, 11, 13, 17, 19, 23]

Сумма: 98

Среднее значение: 14.00

Пример 11: Простые только по главной диагонали

Ввод:

3 3

2 4 6

8 3 10

12 14 5

Вывод:

Ядро простых чисел: (0, 0) до (2, 2)

Простые числа ядра (отсортированы): [2, 3, 5]

Сумма: 10

Среднее значение: 3.33

Решение задачи:

```
def is_prime(n):
    # Проверка числа на простоту
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

def bubble_sort(lst):
    # Пузырьковая сортировка
    n = len(lst)
    for i in range(n):
        for j in range(n - i - 1):
            if lst[j] > lst[j + 1]:
                lst[j], lst[j + 1] = lst[j + 1], lst[j]
    return lst

def find_prime_core(matrix):
    n = len(matrix)
    m = len(matrix[0])
    rows_with_primes = set()
    cols_with_primes = set()
    primes_positions = set()

    # Сначала определим, где находятся простые числа и соберем координаты
    for i in range(n):
        for j in range(m):
            if is_prime(matrix[i][j]):
                rows_with_primes.add(i)
                cols_with_primes.add(j)
                primes_positions.add((i, j))  # Сохраняем координаты простых чисел

    if not rows_with_primes or not cols_with_primes:
        return None, None, None

    min_row = min(rows_with_primes)
    max_row = max(rows_with_primes)
    min_col = min(cols_with_primes)
    max_col = max(cols_with_primes)

    # Собираем простые числа из ядра, не проверяя их снова
    primes = [
        matrix[i][j]
        for i in range(min_row, max_row + 1)
        for j in range(min_col, max_col + 1)
        if (i, j) in primes_positions
```

```
    ]

    return primes, [min_row, min_col], [max_row, max_col]

def print_core_info(primes, top_left, bottom_right):
    print(f"Ядро простых чисел: ({top_left[0]}, {top_left[1]}) до ({bottom_right[0]},
{bottom_right[1]})")
    sorted_primes = bubble_sort(primes.copy())
    print(f"Простые числа ядра (отсортированы): {sorted_primes}")
    total = sum(sorted_primes)
    avg = total / len(sorted_primes)
    print(f"Сумма: {total}")
    print(f"Среднее значение: {avg:.2f}")

def main():
    n, m = map(int, input("Введите размеры матрицы: ").split())
    print("Введите элементы матрицы построчно:")
    matrix = [list(map(int, input().split())) for _ in range(n)]

    primes, top_left, bottom_right = find_prime_core(matrix)

    if primes is None:
        print("Простых чисел не найдено.")
    else:
        print_core_info(primes, top_left, bottom_right)

if __name__ == "__main__":
    main()
```

Методические рекомендации по решению задачи

«Ядро простых чисел в матрице»

1. Общие положения

Цель задачи:

- Освоить алгоритмы работы с двумерными массивами и матрицами.
- Развить навыки создания и использования функций в Python.
- Применить алгоритмы проверки чисел на простоту и сортировки.

Ожидания от участников:

1. Умение работать с матрицами и вложенными списками.
2. Реализация функции для проверки числа на простоту.
3. Создание функций для сортировки методом пузырька и поиска ядра.

4. Навыки обработки данных, ввода/вывода и форматирования результатов.
5. Корректная организация кода с использованием функций.

2. Методика оценки задания

Критерии оценки:

Корректность выполнения всех этапов задачи:

- Ввод и формирование матрицы;
- Корректная проверка чисел на простоту;
- Поиск ядра и правильное определение его границ;
- Сортировка простых чисел в ядре методом пузырька;
- Расчёт суммы и среднего значения.

Обработка ситуаций отсутствия простых чисел.

Структурированность программы:

- Реализация всех действий через функции;
- Чистота и читаемость кода.

Соответствие программы предъявляемым требованиям.

3. Разбор возможных трудностей и типичных ошибок

Возможные трудности при подготовке:

1. Ошибки при работе с матрицей

- Неправильное создание матрицы — вместо использования вложенных списков учащиеся могут создавать одномерный список или вообще вводить данные поэлементно без структуры матрицы.
- Неверный порядок индексов при обращении к элементам матрицы — путаница строк и столбцов (n и m).
- Ошибки при построчном вводе данных — неправильный парсинг строк, игнорирование разделителей.
- Отсутствие проверки на корректность ввода размеров матрицы или её элементов (например, отрицательные числа или нули).

2. Ошибки в реализации функции проверки простого числа (is_prime)

- Игнорирование граничных случаев (например, числа 1 и 2).
- Неверно реализованная проверка на делимость — пропуск проверки на деление на 2 или перебор всех чисел до n вместо $\sqrt{n}$.
- Попытки использовать сторонние библиотеки, хотя они запрещены в условиях задания.

- Преждевременный выход из функции, из-за чего проверяются не все делители.
- Ошибка в условии цикла, которая приводит либо к лишним проверкам, либо к некорректной работе функции.

3. Ошибки в сортировке пузырьком (bubble_sort)

- Попытки использовать встроенные методы .sort() или sorted().
- Ошибка в реализации алгоритма: например, отсутствие вложенного цикла или неправильные границы проходов.
- Отсутствие оптимизаций, когда можно завершить сортировку раньше, если обменов не было.
- Работа с копией списка вместо исходного (если задача требует сортировать "на месте").
- Нарушение порядка элементов при равных значениях (нестабильная сортировка).

4. Ошибки при поиске ядра простых чисел (find_prime_core)

- Неправильное определение строк и столбцов, содержащих простые числа.
- Перепутанные минимальные и максимальные индексы строк/столбцов при определении ядра.
- Повторное вычисление простоты чисел в ядре вместо использования ранее сохранённых результатов.
- Игнорирование проверки ситуации, когда матрица вообще не содержит простых чисел (должно возвращаться None, None, None).
- Ошибка при извлечении чисел из подматрицы — использование неверных границ или пропуск нужных ячеек.

5. Ошибки в организации программы

- Весь код размещён внутри функции main без использования вспомогательных функций.
- Много дублирующегося кода, который можно было вынести в отдельные функции.
- Слишком длинные функции, которые сложно читать и проверять.
- Отсутствие комментариев и аннотаций типов, из-за чего программа плохо читается.
- Неверная структура программы — например, перемешаны ввод, логика и вывод.

6. Отсутствие обработки отсутствия простых чисел

- Нет проверки на пустоту списка простых чисел перед попытками сортировки или вычисления среднего значения.
- Отсутствие корректного вывода сообщения «Простых чисел не найдено».
- Попытка деления на ноль при расчёте среднего значения, если список пустой.

- Некорректное завершение программы при отсутствии простых чисел (например, аварийное завершение с ошибкой).

7. Ошибки в расчётах

- Неверный подсчёт суммы или среднего значения (например, не учитываются все элементы или сумма неправильно считается).
- Отсутствие округления среднего значения до двух знаков после запятой.
- Ошибки в форматировании чисел при выводе результата (например, отсутствие знаков после запятой или их избыток).

8. Ошибки в интерфейсе программы

- Отсутствие понятных приглашений для ввода данных.
- Слишком скупые или, наоборот, избыточные сообщения, которые сбивают пользователя с толку.
- Сообщения об ошибках или результатах некорректно оформлены, не соответствуют примерам.

Кейс №2 «3D-моделирование и 3D-печать»

Задание 1.

Построить 3D-модель детали согласно чертежу, представленному на рисунке 1.

Задание 2.

Постройте 3D-модель детали согласно чертежу, представленному на рисунке 2.

Задание 3.

Создайте сборку из созданных ранее деталей согласно чертежу, представленному на рисунке 3 и спецификации на рисунке 4, используйте стандартные изделия, представленные во встроенной библиотеке, выбранной САПР.

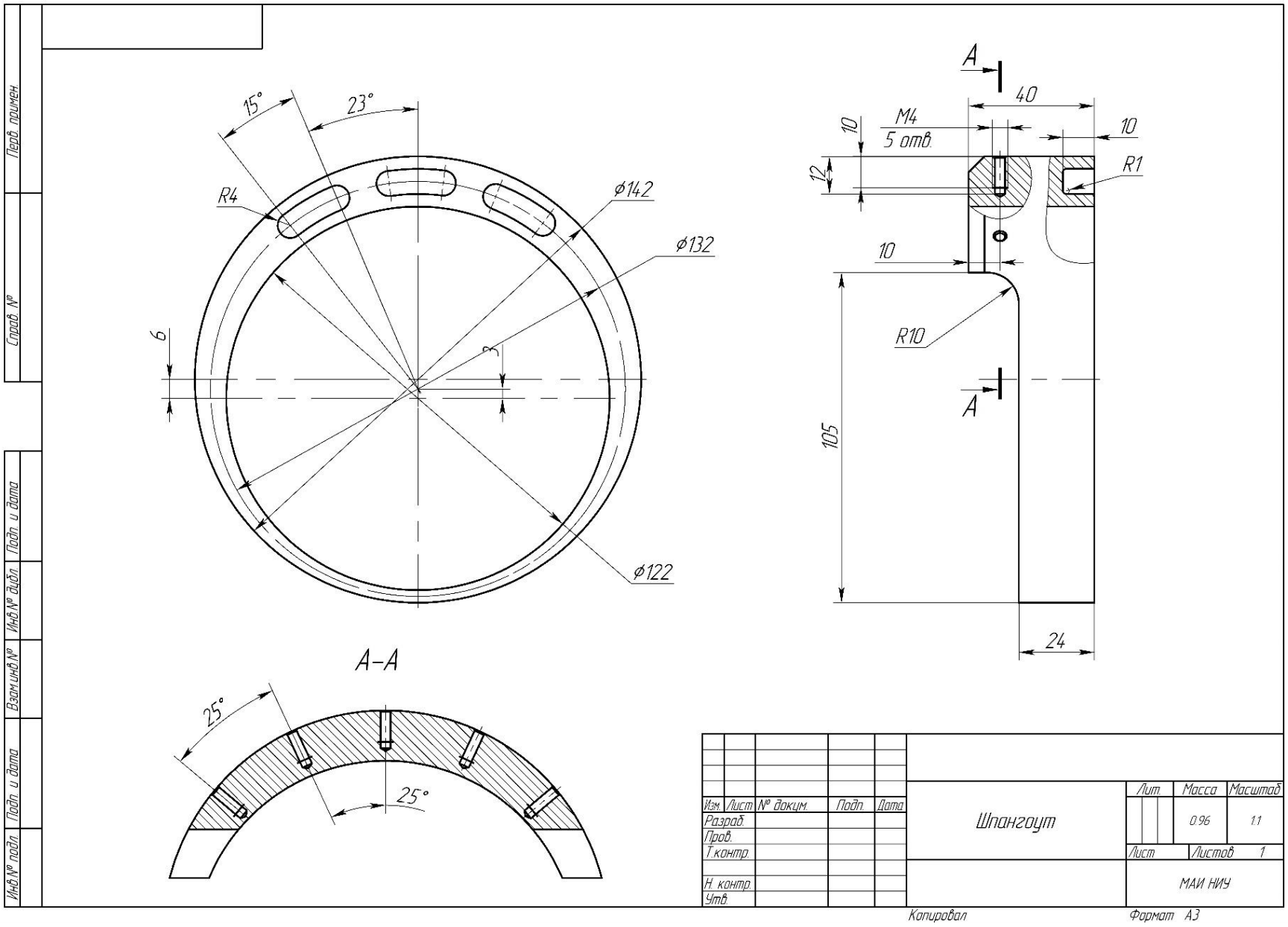

15°
23°
R4
ø142
ø132
ø122
6
3
A
40
M4
5 отв.
10
10
12
R1
10
R10
105
24
A–A
25°
25°
Перв. примен
Справ. №
Подп. и дата
Инв.№ дубл.
Взам.инв.№
Подп. и дата
Инв.№ подл.
Изм. Лист № докум. Подп. Дата
Разраб.
Пров.
Т.контр.
Н. контр.
Утв.
Шпангоут
Лит.
Масса
Масштаб
0.96
1:1
Лист
Листов 1
МАИ НИУ
Копировал
Формат А3

Рисунок 1.

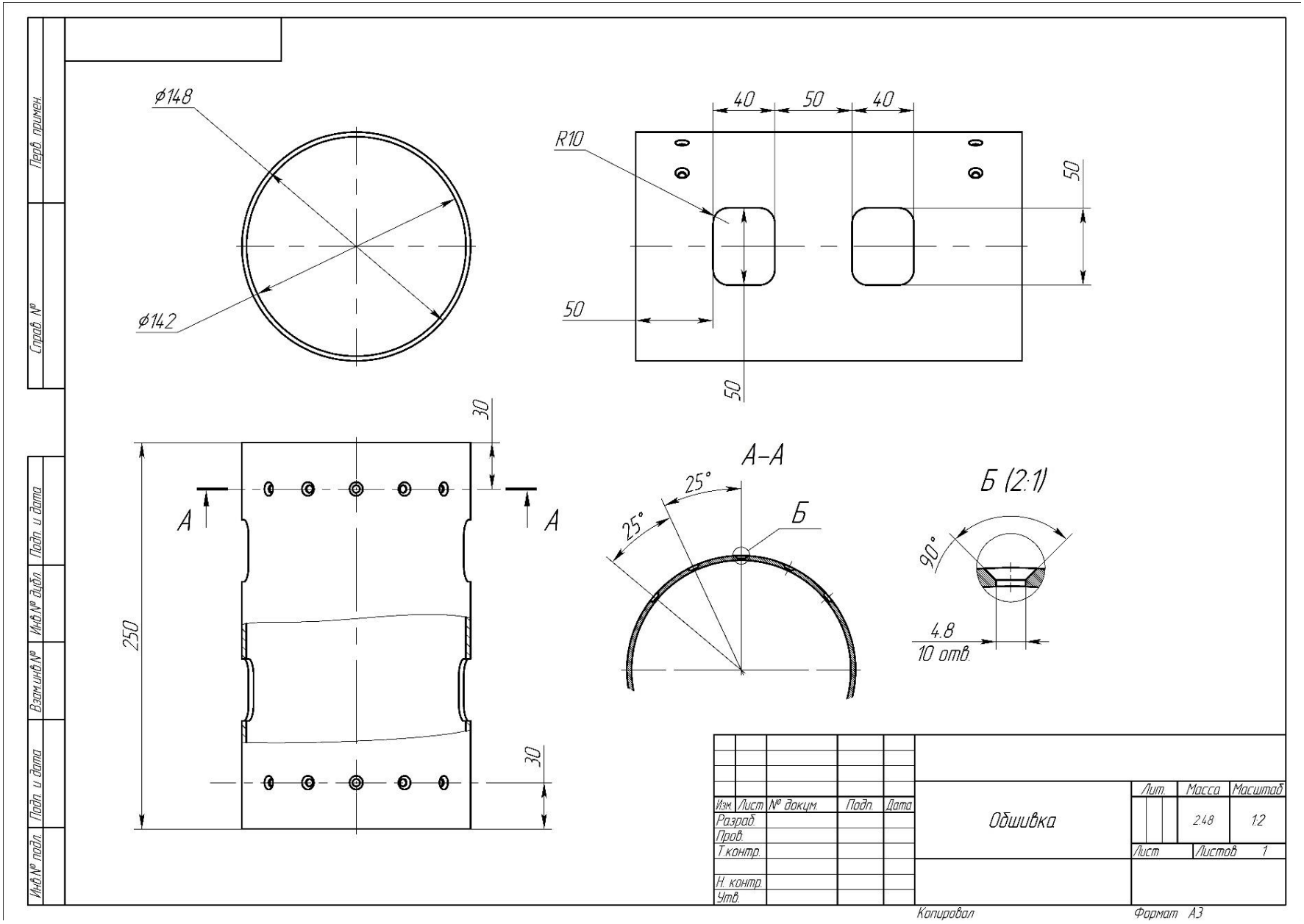

ϕ148
ϕ142
40
50
40
R10
50
50
50
30
250
30
A
A
А–А
25°
25°
Б
Б (2:1)
90°
4.8
10 отв.
Перв. примен
Справ. №
Подп. и дата
Инв.№ дубл.
Взам.инв.№
Подп. и дата
Инв.№ подл.
Изм. Лист № докум. Подп. Дата
Разраб.
Пров.
Т.контр.
Н. контр.
Утв.
Обшивка
Лит.
Масса
Масштаб
248
1:2
Лист
Листов 1
Копировал
Формат А3

Рисунок 2.

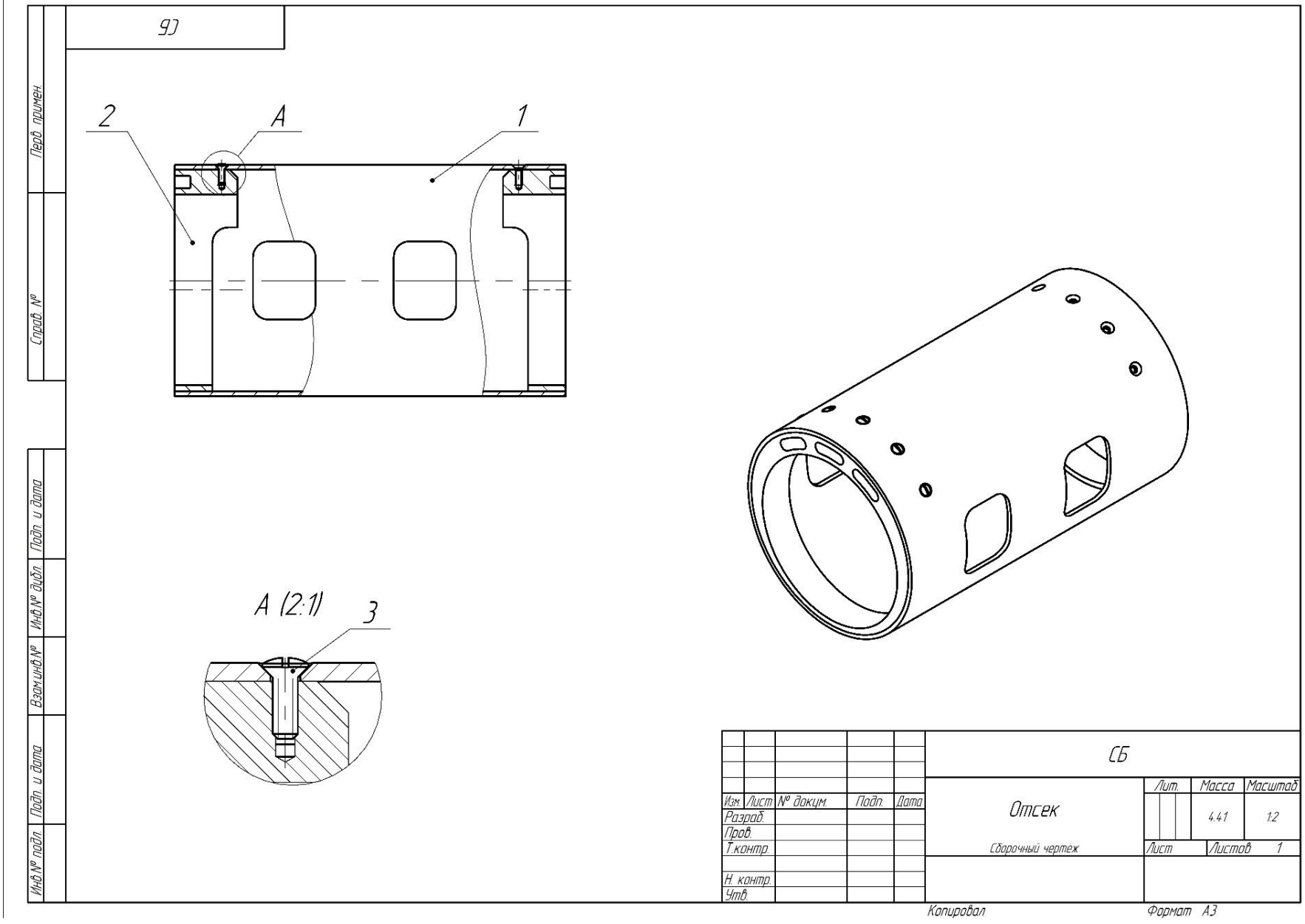
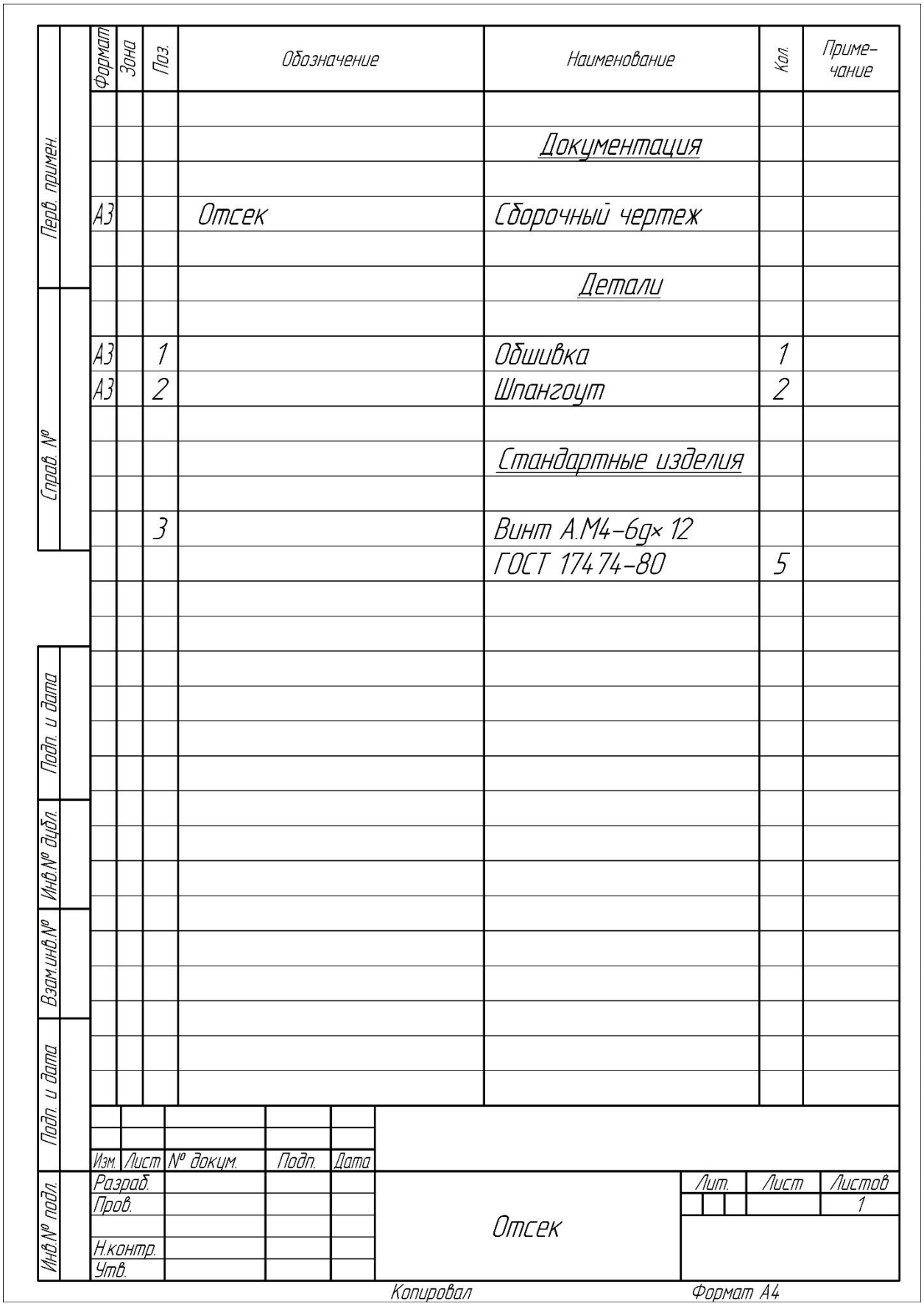

9)
2
A
1
A (2:1)
3
Перв. примен.
Справ. №
Подп. и дата
Инв.№ дубл.
Взам.инв.№
Подп. и дата
Инв.№ подл.
СБ
Отсек
Сборочный чертеж
Лит.
Масса
Масштаб
4.41
1:2
Лист
Листов 1
Изм. Лист № докум. Подп. Дата
Разраб.
Пров.
Т.контр.
Н. контр.
Утв.
Копировал
Формат А3

Рисунок 3.



Формат	Зона	Поз.	Обозначение	Наименование	Кол.	Приме-чание
				Документация		
А3			Отсек	Сборочный чертеж		
				Детали		
А3		1		Обшивка	1	
А3		2		Шпангоут	2	
				Стандартные изделия		
		3		Винт А.М4-6g× 12		
				ГОСТ 17474-80	5	

Изм.	Лист	№ докум.	Подп.	Дата			
Разраб.					Отсек	Лит.	Лист / Листов 1
Пров.							
Н.контр.							
Утв.							

Перв. примен. | Справ. № | Подп. и дата | Инв.№ дубл. | Взам.инв.№ | Подп. и дата | Инв.№ подл.

Копировал | Формат А4

Рисунок 4.

Пример выполнения:

Рисунок 5 – Пример выполнения задания 1.1.

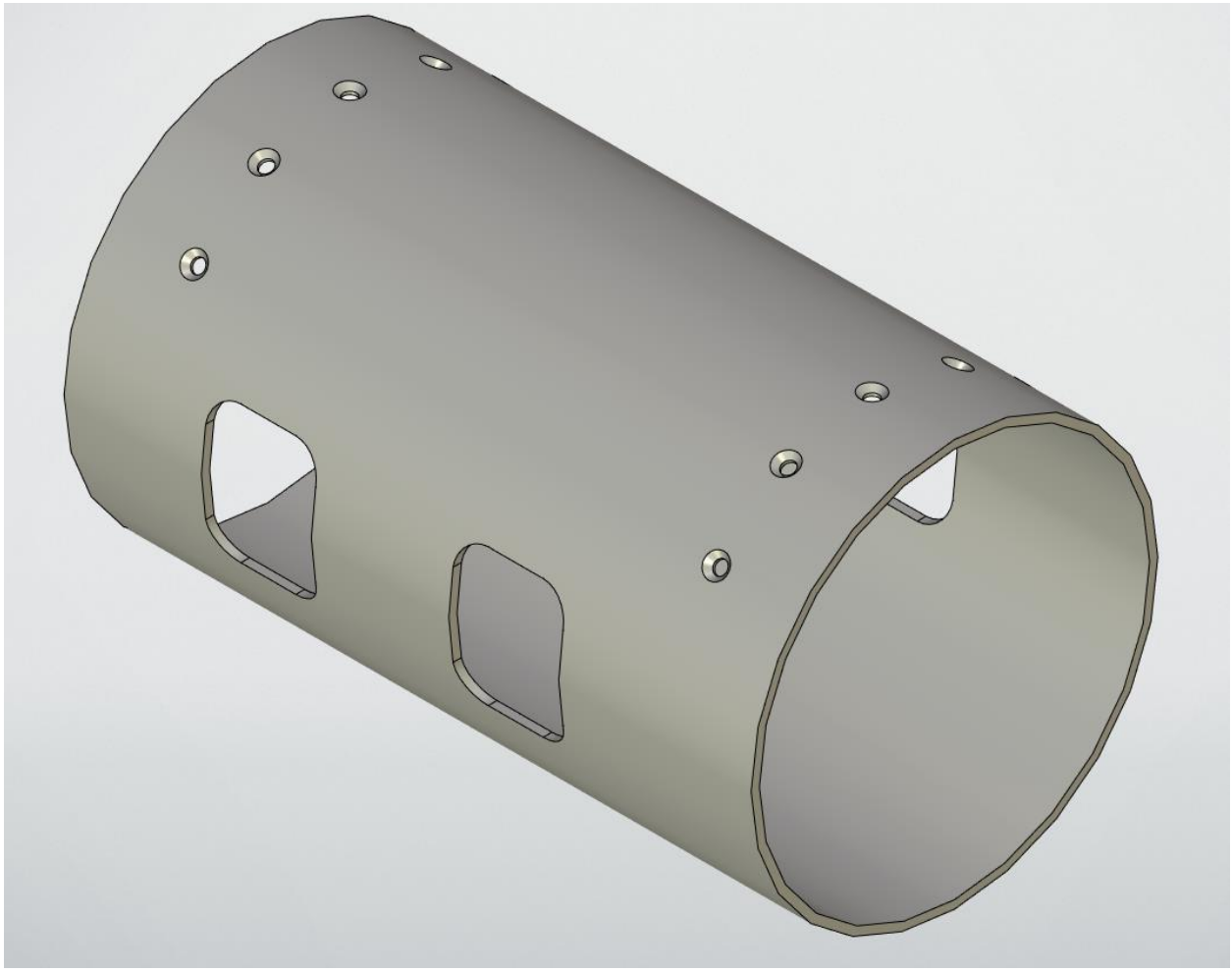


Рисунок 6 – Пример выполнение задания 1.2.

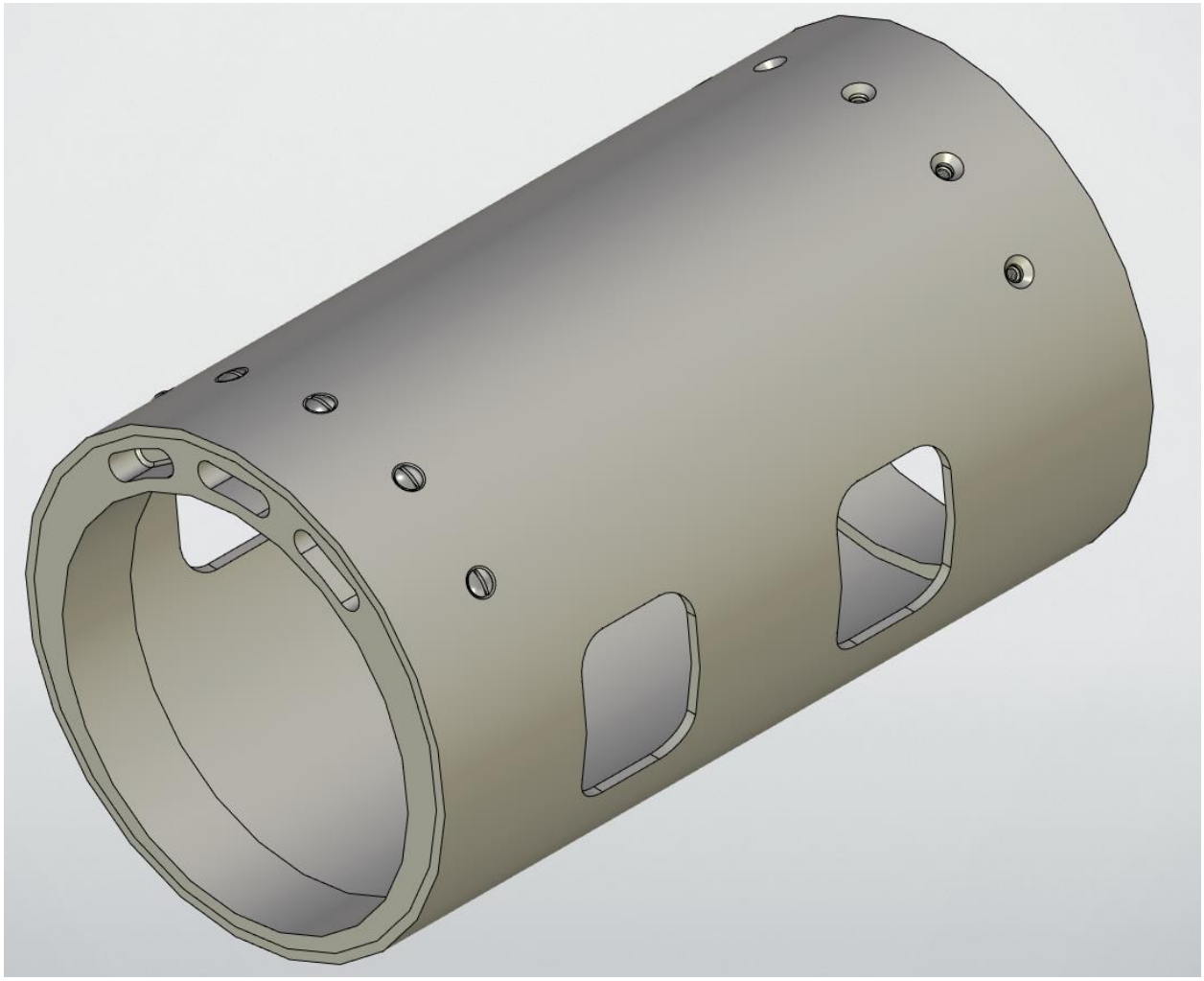


Рисунок 7 – Пример выполнения задания 1.3.

Для успешного выполнения задания экзаменуемый должен ознакомиться со следующими темами инженерных дисциплин:

САПР: системы автоматизированного проектирования, 2D эскиз (набросок, sketch), бобышки (тела, body): вытянутые, повёрнутые, «по сечениям», скругления (сглаживания) и фаски, отверстия, массивы объектов (линейный, круговой); сборки, использование стандартных изделий.

Для построения деталей при выполнении экзаменационного задания с помощью САПР, следует придерживаться следующей последовательности действий:

1. Общая оценка всей детали с целью мысленного разбиения её на отдельные тела, вырезы или другие операции;
2. Построение 2D эскиза для одного тела или выреза;
3. Построение тела или выреза на основе данного 2D эскиза;
4. Повторения шагов 2 и 3 для каждого объекта, определённого на 1ом шаге.

Результатом выполнения задания является файлы в формате соответствующим выбранной САПР, количество которых для каждого задания:

- Задание 1 – 1 файл детали;

- Задание 2 – 1 файл детали;
- Задание 3 – 1 файл сборки.

Таким образом, если экзаменуемый выполнил все 3 задания, то на проверку он должен отправить всего 3 файла.

Особое внимание требуется уделить построение 2D эскизов. Как правило, именно его размеры и геометрические взаимосвязи служат надежным инструментом при однозначном определении формы и размеров деталей. При простановке размеров эскиза следует использовать принцип «от базы», когда за базовую принимается, как правило, самая длинная линия в данном измерении и размеры, как правило, проставляют от нее. На рисунке 1 приведен пример простановки размеров «от базы» в горизонтальном направлении.

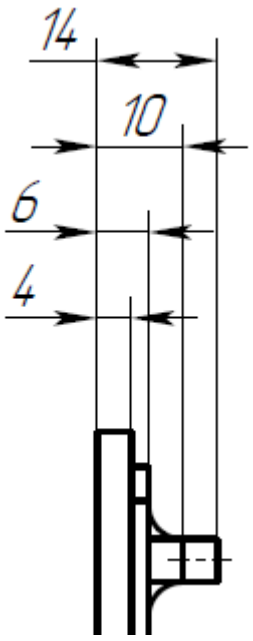


Рисунок 8 – Пример простановки размеров

При построении эскиза рекомендуется сначала создать его примерную форму с помощью прямых линий и дуг, затем добавить геометрические взаимосвязи, уточнив его форму, а затем образмерить его с помощью размеров, указанных на чертежах.

При выполнении заданий на построение деталей, следует учитывать, что чертежи содержат полную информацию о форме и размерах деталей. При построении деталей возможна любая ориентация их в пространстве.